

▼ EJ 2 - PANDAS

Un importante servicio de monitoreo de aplicaciones cloud, sufrió un incidente en las últimas semanas, tras el cual está conduciendo una investigación del impacto que continua teniendo en sus integraciones a nivel de conectividad. Como analista externos, tu misión es:

A) Calcular las métricas sumarizadas 'handshake_failed', 'ssl_failed', 'tls_failed' para cada uno de los integration_id a partir del timestamp '1592187602', momento en el cual comenzó el incidente. El formato esperado del df salida es:

```
[ ] # Metricas = ('client_id', 'integration_id' , 'metric', 'timestamp', 'value')
# Integracion = ('integration_id', 'data_provider', 'integration_name')

[ ] #Hago los imports necesarios
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt import matplotlib.patches as mpatches

[ ] # Levanto los archivos '.csv' dados:
Metricas = pd.read_csv('metrics.csv')
Integracion = pd.read_csv('integrations.csv')

[ ] #Filtro para los timestamps mayores a '1592187602'
FiltroTime = Metricas['timestamp'] > '1592187602'
Metricas = Metricas[FiltroTime]

[ ] #Filtro por las metricas solicitadas: 'handshake_failed', 'ssl_failed', 'tls_failed'
FiltroMetricas =( Metricas['metric'] == 'handshake_failed' | Metricas['metric'] == 'ssl_failed' | Metricas['metric'] == 'tls_failed'))
Metricas = Metricas[FiltroMetricas]

[ ] #Metricas = ('client_id', 'integration_id' , 'metric', 'timestamp', 'value')
MetricasPivoteadas = pd.pivot_table(Metricas, index=['integration_id', 'metric'], aggfunc={'value':'sum'}).unstack()

[ ] #Renombro mis columnas como muestra la imagen en el enunciado
MetricasPivoteadas.columns = ['handshake_failed', 'ssl_failed', 'tls_failed']
```

```
[ ] #Relleno con 0 en el caso de haber obtenido un NaN  
MetricasPivoteadas = MetricasPivoteadas.fillna(0)
```

B) Indicar aquellas 5 integraciones que tuvieron la mayor cantidad de errores 'handshake_failed' a partir del inicio del incidente. El formato esperado del df de salida es: (integration_name, tot_handshake_failed)

```
[ ] #Hago de cuenta que no tengo lo de arriba a modo de usar las mismas variables.
```

```
[ ] #Filtro para los timestamps mayores a '1592187602' osea a partir del incidente.  
FiltroTime = Metricas['timestamp'] > '1592187602'  
Metricas = Metricas[FiltroTime]
```

```
[ ] #Filtro por las metrica solicitada: 'handshake_failed'  
FiltroMetricas = Metricas['metric'] == 'handshake_failed'  
Metricas = Metricas[FiltroMetricas]
```

```
[ ] #Junto mis dos Data Frames para poder obtener el integration_name  
#Metricas = ('client_id', 'integration_id', 'metric', 'timestamp', 'value', 'integration_name', 'data_provider')  
MetricasNombradas = Metricas.merge(Integracion, on='integration_id', how='inner')
```

```
[ ] #Agrupo por integration_name y cuento la cantidad de metricas 'handshake_failed'  
Top5Integraciones = MetricasNombradas.groupby('integration_name').agg({'metric': 'count'}).reset_index()
```

```
[ ] #Renombro mis columnas como solicita el ejercicio  
Top5Integraciones.columns = ['integration_name', 'tot_handshake_failed']
```

```
[ ] #Me quedo con los 5 integrations_names con mas 'handshake_failed'  
Top5Integraciones = Top5Integraciones.nlargest(5, columns = 'tot_handshake_failed')
```

▼ EJERCICIO 1 - SPARK

Dado los acontecimientos en USA, deseamos obtener datos que nos den mayor información sobre las muertes de gente de raza negra por parte de oficiales de policía. Para ello, tenemos un csv con información sobre las muertes por parte de oficiales de policía en USA para 2015 hasta 2017: (name, date, race, city, state) Y otro csv con información sobre el porcentaje de pobreza en las ciudades de USA: (state, city, poverty_rate) Se pide:

A) Obtener las 10 ciudades con mayor diferencia entre el porcentaje de pobreza de la ciudad y el porcentaje de pobreza del estado en el que se encuentra esa ciudad. Por ejemplo si en la ciudad de Houston la pobreza es de 15.2 y la pobreza en Texas (el estado donde se encuentra Houston) es de 11.1, la diferencia es 4.1.(15 pts)

```
[ ] #Agrego los imports necesarios
from pyspark.sql import *
from pyspark.sql.functions
import * from pyspark
import SparkContext from pyspark
import SQLContext
import random
```

```
[ ] #Creo la Spark Session
spark = SparkSession.builder.getOrCreate()
```

```
#Creo el Spark Context
sc = spark.sparkContext
```

```
#Genero mis RDDs
PobrezaRDD = sc.parallelize(pobreza)
MuertesRDD = sc.parallelize(muertes)
```

```
[ ] #(state, city, poverty_rate)
#Supongo que en el RDD hay una unica aparicion para cada city (ejemplo: (Texas, Houston, %) aparece una sola vez)
#Entonces ya tengo el % por CITY.
```

```
[ ] #Pobreza = (state, (city, poverty_rate) )
Pobreza = PobrezaRDD.map(lambda x: (x[0], (x[1], x[2])) )

[ ] #Creo un RDD PobrezaEstado = (state, poverty_rate) para pode calcular el total.
PobrezaEstado = PobrezaRDD.map(lambda x: (x[0],x[2]))

[ ] #Reduzo por state para obtener la suma del % de pobreza por ciudad en ese estado.
#(state, total_poverty_rate)
PobrezaEstado = PobrezaEstado.reduceByKey(lambda a,b: a+b)

[ ] #Tengo Pobreza = (state, (city, poverty_rate)) y PobrezaEstado = (state, (total_poverty_rate))

#Hago un Join por state (state, (city, city_poverty_rate, total_poverty_rate) )
PobrezaEstadoCiudad = Pobreza.join(PobrezaEstado).cache()

[ ] #Mapeo mi RDD PobrezaEstadoCiudad (state, city, diferencia_poverty)
PobrezaEstadoCiudad = PobrezaEstadoCiudad.map(lambda x: (x[0], x[1][0], x[1][1]- x[1][2]) )

[ ] #Me quedo con las 10 ciudades con mayor diferencia.
Top10MasDiferencia = PobrezaEstadoCiudad.takeOrdered(10, lambda x: -x[2])
```

B) Obtener la cantidad de muertes de gente de raza negra por parte de oficiales de policía, agrupada por estados que comparten el mismo nivel de pobreza redondeado al entero más cercano. Por ejemplo, si NJ tiene una pobreza de 10.33, AL una de 20.64 y AZ una de 10.44, NJ y AZ quedarían juntos representados por el nivel de pobreza de 10 y AL en otro grupo con el nivel 21. La salida debe tener el formato: (nivel_de_pobreza, total_de_muertes) (15 pts)

```
[ ] #Organizo mi RDD Muertes para contar la cantidad de muertos negros por estado.  
#(name, date, race, city, state)  
#(state, 1 si es negro 0 si no lo es )  
Muertes = MuertesRDD.map(lambda x: (x[4] , 1 if x[2] == 'black' else 0 ))  
  
[ ] #Cuento la cantidad de muertes de negro por estado  
#(state, total_negros_muertos)  
MuertesNegros = Muertes.reduceByKey(lambda a,b: a+b)  
  
[ ] #Voy a utilizar el RDD hecho en el item anterior PobrezaEstado = (state, poverty_total)  
#Ahora tengo MuertesNegros = (state, total_negros_muertos)  
  
#Junto ambos RDD (state, (poverty_total, total_negros_muertos))  
MuertesyPobreza = PobrezaEstado.join(MuertesNegros)  
  
[ ] #Organizo mi RDD para redondear el % y eliminar el state.  
#(round_poverty_rate, total_negros_muertos)  
MuertesyPobreza = MuertesyPobreza.map(lambda x: (round(x[1][0]), x[1][1]))  
  
[ ] #Agrupo por % de poverty_rate y sumo la cantidad de muertes de negros para ese %  
MuertesyPobreza = MuertesyPobreza.reduceByKey(lambda a,b: a+b)
```

EJ (3)

A)

- TENGO 8 VECTORES EN 7 DIMENSIONES
- EL MH ES UNA DIMENSION, 0, 1, 5 SON LOS 3 MH.
- TENEMOS $D=3$ Y $B=1$ OSEA 1 TABLA Y 3 MH POR TABLA

M.	V1	V2	V3	V4	V5	V6	V7	V8
MH1	0	1	1	0	1	1	0	0
MH2	1	0	1	1	1	1	0	0
MH3	0	0	1	0	1	0	0	1

BUENO EL PROBLEMA ACA ES QUE ESTA TABLA ME MANDA LOS VECTORES A POSICIONES DE UNA TABLA DE 8 POSICIONES (POQUE TENDO 3 BITS DE DIRECCIONAMIENTO) ENTONCES COMO VOY TENDO UNA TABLA DE 6 ENTRADAS EN DONDE ENTRAEN 4 VALORES, VOY A TENER QUE HASHEAD ESTOS VALORES PARA QUE ENTRAEN EN MI TABLA. PARA ESTO UTILIZO UNA FUNCION DE HASHING UNIVERSAL PARA VALORES ATOMICOS:

$$H(x) = (Ax + B \bmod p) \bmod m$$

LO IMPORTANTE ACA ES QUE $m=6$ YA QUE MI ESPACIO DE DIRECCIONES ES DE 6.

- $m=6$ LOS VALORES DE A Y B DEBEN
- $p=7$ CUMPLIR QUE NO TENGA MAS DE 4 COLISIONES POR BUCKET.

$$H(x) = (Ax + B \bmod 7) \bmod 6$$

$$v_1 = 010 = 2$$

$$v_5 = 111 = 7$$

$$v_2 = 100 = 4$$

$$v_6 = 110 = 6$$

$$v_3 = 111 = 7$$

$$v_7 = 000 = 0$$

$$v_4 = 010 = 2$$

$$v_8 = 001 = 1$$

PROUEBO CON $A=1$ $B=2$.

$$H(x) = (x+2 \bmod 7) \bmod 6$$

$$\circ H(2) = (2+2 \bmod 7) \bmod 6 = 4$$

$$\circ H(4) = (4+2 \bmod 7) \bmod 6 = 0$$

$$\circ H(7) = (7+2 \bmod 7) \bmod 6 = 2$$

$$\circ H(2) = 4$$

$$\circ H(6) = (6+2 \bmod 7) \bmod 6 = 1$$

$$\circ H(3) = 2$$

$$\circ H(0) = (2 \bmod 7) \bmod 6 = 2$$

$$\circ H(1) = (1+2 \bmod 7) \bmod 6 = 3$$

V2	V6	V3	V8	V1	
V7 Q		V5		V4	
0	1	2	3	4	5

DE ESTA MANERA RESPETA LA CONDICIÓN
DADA DE 6 REGISTROS CON 4 POS BUCKET

B)

AHORA TENGO $Q = [1 \ 0 \ 1 \ 0 \ 0 \ 1]$

	VQ
MH1	1
MH2	0
MH3	0

ESTO ES POS 100 = 4

LE APLICO MI FUNCIÓN $H(x)$

$$H(4) = (4 + 2 \text{ MOD } 7) \text{ MOD } 6$$

$$H(4) = (6 \text{ MOD } 7) \text{ MOD } 6$$

$$H(4) = 0$$

UBICO MI QUERY EN LA TABLA DE ARRIBA

ENTONCES LOS VECTORES CERCAOS A MI QUERY SON V2 y V7.

C) PARA ESTE PUNTO USARIA UNA FUNCIÓN DE HASHING PERFECTO Y MINIMO. ESTA FUNCIÓN ME ASEGURA O(1) EN CONSULTAS O(M) EN ESPACIO.

ENTONCES PARA MIS 8 CLAVES NO ES NECESARIO UNA TABLA DE 24 REGISTROS SIMPLEMENTE UNA DE 8. LA FUNCIÓN DE HASHING PERFECTA Y MINIMA MAS USADA ES HDC.

EJ (4)

- CANTIDAD DE CLICKS QUE HAN HECHO USUARIOS SOBRE AVISOS.

17 CIUDADES (FILAS) } MATRIZ DE
 1500 AVISOS (COLUMNAS) } 17×1500
 \downarrow
 CELDA CANT DE CLICKS.

A) QUIERO AGRUPAR EN 4 CATEGORIAS USANDO LA DVS

1) PARA PODER AGRUPAR EN 4 CATEGORIAS TENGO QUE REDUCIR LA MATRIZ ORIGINAL A RANGO 4. PARA ESTO VOY A UTILIZAR PCA QUE ESEQUIVALENTE (POD TEOREMA) A DVS.

$$X = U \Sigma V^T \text{ PERO LO REDUZCO A}$$

$$K = 4$$

$$\hat{X} = U_R \Sigma_R V_R^T \quad U_R = 17 \times 4$$

$$\Sigma_R = 4 \times 4$$

$$V_R^T = 4 \times 17$$

PARA OBTENER TODO ESTO DEBO CALCULAR AUTO VECTORES Y AUTOVALORES DE $X^T X$

AHORA ESTAS 3 MATRICES HACEN REFERENCIA A:

U_R = INDICA LA RELACIÓN ENTRE CIUDADES CON LAS 4 CATEGORÍA DE AVISOS.

Σ_R = INDICA LA IMPORTANCIA DE CADA CATEGORÍA.

V_R^T = INDICA LA RELACIÓN DE LOS AVISOS CON LAS 4 CATEGORÍAS.

A SU VEZ SABEMOS QUE EN PCA

$$X = U \Sigma V^T$$

$$XV = U \Sigma V^T V$$

ID DOB UNIDAD

$XV = U \Sigma$ DONDE LAS 4 COLUMNAS DE V SON LAS COMPONENTES PRINCIPALES PARA CADA CATEGORÍA.

B) PARA PODER REPARTIR LOS US 100 000 DE VISAS EN QUE CIUDADES HARÁN MÁS CLICKS Y EN QUÉ AVISOS, PARA ESTO SE PUEDEN VER LOS VALORES DE LAS CELDAS DE V .

LOS DIRECTIVOS TENDRÍAN QUE TENER EN CUENTA LA ENERGÍA DE LA MATRIZ, PARA SABER SI ES UN K QUE APROXIMA BIEN:

$$\sum_{i=1}^n \sigma_i^2 = \text{ENERGÍA}$$

ENTONCES CON ESTO CALCULAMOS:

$$\alpha = \frac{\sum_{i=1}^k \sigma_i^2}{\sum_{i=1}^N \sigma_i^2} \quad k = 4 \text{ (REDUCCIÓN)}$$

N = LA CANT AUTOVAL.
QUE ES 17.

ENTONCES SI GRAFICAMOS PARA LOS DISTINTOS
K PODEMOS DETECTAR POSIBLES 'CODOS'
PERSONALMENTE LES RECOMIENDO QUE HAGAN
ESTE GRAFICO DADA QUE VERIFIQUEN SI SU
K NO PRODUCE SALTOS.